CLAIMS

1. A method for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before output queues, the method comprising the steps of:

receiving a stream of data from the switching fabric;

subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected).

Sub A1

2. A method for bandwidth scheduling according to claim 1, wherein the stream of data includes identifiable data packets;

subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

3. A method for bandwidth scheduling according to claim 2, wherein each packet contains information about its flow identity, namely port (number) and traffic class.

4. A method for bandwidth scheduling according to claim 3, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

5. A method for bandwidth scheduling according to claim 4, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

6. A method for bandwidth scheduling according to claim 5, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

7. A method for bandwidth scheduling according to claim 4, wherein a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class,

8. A method for bandwidth scheduling according to claim 7, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max}$.

9. A method for bandwidth scheduling according to claim 8, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

10. A method for bandwidth scheduling according to claim 3, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, wherein,

for a traffic class having the ratio TC/TCP_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

11. A method for bandwidth scheduling according to claim 3, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

12. A method for bandwidth scheduling according to claim 11, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

13. A method for bandwidth scheduling according to claim 10, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes.

14. A method for bandwidth scheduling according to claim 13, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the configuration parameter WTC.

15. A method for bandwidth scheduling according to claim 3, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class.

16. A method for bandwidth scheduling according to claim 15, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class with the ratio BL/BLP_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

17. A method for bandwidth scheduling according to claim 16, wherein the BL counter is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter remains at the lower limit and the counter BL is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

18. A method for bandwidth scheduling according to claim 3, wherein, if one

Snap

traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

19. A method for bandwidth scheduling according to claim 18, wherein, when a packet is accepted in a traffic class having the maximum accepted bandwidth ($TC = TCP_{max}$), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length × give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the charity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length × WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

20. A method for bandwidth scheduling according to claim 3, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

21. A method for bandwidth scheduling according to claim 3, wherein flows are grouped together by means of a hash function into a set of flow groups.

22. A method for bandwidth scheduling according to claim 21, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio $FG/FG_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

23. A method for bandwidth scheduling according to claim 22, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a

virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

24. A method for bandwidth scheduling according to claim 6, 9, 10, 12, 14, 17, or 19, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio $FG/FG_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

25. A method for bandwidth scheduling according to claim 23, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise

if Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise;

accept.

26. A method for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located after output queues, the method comprising the steps of:

Sub B1

0007464954600

receiving a stream of data including identifiable data packets from the switching fabric;

subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, and a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

27. A method for bandwidth scheduling according to claim 26, wherein:

a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

28. An arrangement for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before output queues, the arrangement comprising:

means for receiving a stream of data from the switching fabric;

means for subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected).

29. An arrangement for bandwidth scheduling according to claim 28, wherein the stream of data includes identifiable data packets; and further comprising

means for subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

30. An arrangement for bandwidth scheduling according to claim 29, wherein each packet contains information about its flow identity, namely port (number) and traffic class.

31. An arrangement for bandwidth scheduling according to claim 30, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

Sub B7

32. An arrangement for bandwidth scheduling according to claim 31, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

33. An arrangement for bandwidth scheduling according to claim 32, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

Sub A1

34. An arrangement for bandwidth scheduling according to claim 31, wherein a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class,

35. An arrangement for bandwidth scheduling according to claim 34, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max}$.

36. An arrangement for bandwidth scheduling according to claim 35, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

37. An arrangement for bandwidth scheduling according to claim 30, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, and wherein, for a traffic class having the ratio TC/TCP_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

38. An arrangement for bandwidth scheduling according to claim 30, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

39. An arrangement for bandwidth scheduling according to claim 38, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

40. An arrangement for bandwidth scheduling according to claim 37, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes.

41. An arrangement for bandwidth scheduling according to claim 40, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the

configuration parameter WTC.

42. An arrangement for bandwidth scheduling according to claim 30, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class.

Sub A1

43. An arrangement for bandwidth scheduling according to claim 42, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class with the ratio BL/BLP_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

44. An arrangement for bandwidth scheduling according to claim 43, wherein the counter BL is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter will remain at the lower limit and the counter BL is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

45. An arrangement for bandwidth scheduling according to claim 30, wherein, if one traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

46. An arrangement for bandwidth scheduling according to claim 45, wherein, when a packet is accepted in a traffic class having the maximum accepted bandwidth ($TC = TCP_{max}$), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length × give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the charity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length × WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

47. An arrangement for bandwidth scheduling according to claim 30, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic

class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

48. An arrangement for bandwidth scheduling according to claim 30, wherein flows are grouped together by means of a hash function into a set of flow groups.

49. An arrangement for bandwidth scheduling according to claim 48, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio FG/FG_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

50. An arrangement for bandwidth scheduling according to claim 49, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

51. An arrangement for bandwidth scheduling according to claim 33, 36, 37,

Sub A' Cont.

39, 41, 44, or 46, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio FG/FG_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

52. An arrangement for bandwidth scheduling according to claim 50, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise

if Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise;

accept.

53. An arrangement for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located after output queues, the arrangement comprising:

means for receiving a stream of data including identifiable data packets from the switching fabric;

means for subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, and a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

54. An arrangement for bandwidth scheduling according to claim 53, wherein:

a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

Sub A1

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.